IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Application of: Dettinger et al.	§	Filed: October 9, 2003
Serial No.: 10/682,133	§	Group Art Unit: 2166
Confirmation No.: 1361	§	Examiner: Khanh B. Pham

For: MODELING AND IMPLEMENTING COMPLEX DATA ACCESS OPERATIONS BASED ON LOWER LEVEL TRADITIONAL OPERATIONS

MAIL STOP APPEAL BRIEF - PATENTS
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

CERTIFICATE OF MAILING OR TRANSMISSION

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Mail Stop Appeal Brief - Patents, Commissioner for Patents, P. O. Box 1450, Alexandria, VA 22313-1450, or facsimile transmitted to the U.S. Patent and Trademark Office to fax number 571-273-8300 to the attention of Examiner Khanh B. Pham, or electronically transmitted via EFS-Web, on the date shown below:

June 30, 2008 /Tammi Thomas/
Date Tammi Thomas

REPLY BRIEF

Dear Sir:

Applicants submit this Reply Brief to the Board of Patent Appeals and Interferences in response to Examiner's Answer dated April 30, 2008. While Applicants' maintain each of the arguments submitted in Applicants' previously submitted Appeal Brief, Applicants make the following further arguments in light of the Examiner's Answer. Please charge any additional fees that may be required to make this Reply Brief timely and acceptable to Deposit Account No. 09-0465/ ROC920030237US1.

ARGUMENTS

Scalzo Does Not Anticipate Independent Claims 1, 26, or 50

The Examiner continues to suggest that *Scalzo* discloses the method recited by claim 1 for "managing execution of query operations in a data processing system." Applicants respectfully disagree. Claim 1 recites, in part:

> executing the initial query operation;
> determining an operation status of the initial query operation;
> selecting one of the plurality of subsequent query operations based on the operation status;
> performing the selected subsequent query operation;
> updating the operation status based on a result of the subsequent query operation;
> managing execution of any remaining subsequent query operations on the basis of the updated operation status; and

Claims 26 and 50 recite a similar limitation. As claimed, the method includes determining an operation status, selecting a second operation to perform based on the operation status, and also includes, after performing the second operation, updating the operation status and managing additional operations on the basis of the updated operation status. That is, after performing an initial operation, an operation status is determined. Based on the status, a second operation is selected and performed. Thereafter, the operations status is updated a second time, and additional operations are performed, on the basis of the second update. Thus, the claim clearly specifies that the operation status is updated twice, a first time following the "execution of the initial query operation", and a second time following the performance of "the selected subsequent query operation."

The Examiner relies on an implementation of an "upsert" operation disclosed in *Scalzo* to suggest that this reference anticipates claims 1, 26, and 50. As the Examiner correctly describes, an "upsert" operation is a combination of two specific query operations – an update and insert operation. Specifically, a first database operation of an "update" is performed, and if it fails (because there is not an existing record to update), a second database operation of insert is performed (adding a new record to the

database). More simply, the "upsert" operation follows a "first try A, then try B" pattern to update, or add, a record in a database. The implementation of this pattern cited by the Examiner includes a "while" loop used to perform the "upsert" operation for multiple records in an input file.

Specifically, the Examiner's Answer includes the following:

> An implementation of the UPSERT in form of C programming languages is shown at pages 7-8, which is a more detail version of appellant's algorithm listed in Table I. A portion of Scalzo's UPSERT Pro-C- program at pages 7-8 is reproduced below:
> **********
> [01] /* Process data file records */
> [02] while (fget (rec, sizeof rec, fid) != NULL) {
> [03] /* First - try to update existing record */
> [04] EXEC SQL EXECUTE update_command
> [05] /* Second - if update fails because record not found, then insert record */
> [06] if (sqlca.sqlcode == 1403)
> [07] EXEC SQL EXECUTE insert_command {
> [08] }
> [09] else if (sqlca.sqlcode != 0)
> [10] }
> **********
> … As seen in the Pro-C program above, Scalzo utilizes a "while" loop to process file records. The expression "fget (rec, sizeof rec, fid) != NULL" when executed results in either "true" or "false", is operation condition which decides when the program will be terminated to exit the while loop. The "fget" function takes three parameters: rec (Record), sizeof rec (size of record), and fid (file 10) and returns a record read from the inputted file. After each iteration of the while loop, the "fget" is executed and returns the next record in the inputted file, the returned record is compared with "NULL" to determine current operation status.

Examiner's Answer, p. 10. Line [04] of this C program includes the update operation, and line [07] includes the insert operation. The Examiner is correct in suggesting that, for each record, an "UPDATE" operation is attempted (line [04]), and if this operation fails, an "INSERT" operation is performed (lines [06]-[08]). However, nothing in this "first try A, then try B" structure does the code cited by the Examiner carry out the claimed steps of

> performing the selected subsequent query operation;
> updating the operation status based on a result of the subsequent query operation; and
> managing execution of any remaining subsequent query operations on the basis of the updated operation status.

In particular, nothing in the C code cited by the Examiner includes "updating the operation status based on the result of the subsequent query operation" and "managing execution of any remaining subsequent query operations on the basis of the updated operation status." Instead, each pass through the "while" loop" executes the "upsert" operation, i.e., the update and insert operations. These operations may be performed multiple times for different records retrieved from an input file using the well known "fgets" command.

The Examiner suggests:

> [*Scalzo* discloses the claimed steps of] selecting one of the plurality of subsequent query operations based on the operation status; performing the selected subsequent query operation" at top of page 8, (Executing insert_command if update fails).

Examiner's Answer, p. 4. That is, the Examiner suggests that performing the "insert" operation anticipates performing the claimed "subsequent query operation." Even assuming, *arguendo*, that the Examiner is correct on this point, no operation status is updated based on the result of the "Insert" operation. Nevertheless, the Examiner tries to suggest that the "while" loop evaluated between instances of performing the "upsert" operation using the call to "fgets" anticipates the claimed steps of "updating the operation status based on a result of the subsequent query operation" and "managing" steps recited by claim 1. Clearly, however, the "fgets" call provided in the standard C library is not a database query operation. Instead, "fgets" provides a well known C library call used to read a specified number of characters from a file. And even if the "fgets" command were read to be an "operation status" for a "query operation" it is not an "operation status [which was updated] based on a result of the subsequent query operation," as recited by the claim. That is, the "fgets" call included in the while loop condition is not performed based on update to an operation status made after performing the "insert" operation (which the examiner argues discloses the claimed "subsequent query operation." Instead, the "while loop" uses the "fgets" command to simply read in the next record from an input file and perform the "upsert" operation on that record, until the last record is read.

Accordingly, Applicants submit that the Examiner is incorrect in suggesting that

> *Scalzo* updates the operation status from "TRUE" to "FALSE" to cause the program to exit the while loop. Otherwise, if the operation status was not changed, the while loop will run indefinitely and the Pro-C program will never be terminated.

Examiner's Answer, p. 10. While the Examiner is correct that the call to "fgets" evaluates to "TRUE" or "FALSE" on each pass through the "while" loop, this evaluation is simply not an "operation status" that was updated "based on a result of the subsequent query operation," as recited by claim 1. First, the "operation status" of the while loop and the "fgets" command are not database query operations. And even if they were, the call to the "fgets" command (used to read a next record from an input file) is not managed, selected, or performed, "based on a result of the subsequent query operation," as recited by the claim, or based on the result of the "insert" operation, as suggested by the Examiner.

Accordingly, for all the foregoing reasons, Applicants submit that independent claims 1, 26, and 50 are believed to be allowable, and allowance of the claims is respectfully requested. Furthermore, claims 2-8, and 27-32 depend upon claims 1 and 26, respectively and Applicants respectfully that the Board vacate this rejection.

Scalzo Does Not Anticipate Independent Claims 9, 33, or 51

Similarly, Applicants submit that *Scalzo* does not teach "managing, using a composite query operations manager, execution of the initial query operation and the plurality of subsequent query operations on the basis of the selection logic and the plurality of failure conditions," as recited in claims 9, 33, and 51.

In rejecting these claims, the Examiner again suggests that the claimed step of executing an "initial query operation" and a "plurality of subsequent query operations" is disclosed by multiple passes through the while loop shown in *Scalzo* pages 7-8. However, as demonstrated above, each pass through the while loop independently performs an "upsert" operation using a record read from an input file.

Applicants respectfully submit that *Scalzo* in no way discloses that each pass through the while loop is performed "on the basis of the selection logic and the plurality of failure conditions," as recited in the present claims. In fact, the cited code does not

disclose performing a “plurality of subsequent query operations” at all. Instead, the while loop is used to read records from an input file using the “fgets” command. On each pass through the loop, a record is read from the input file and the “upsert” operation is performed.

Accordingly, claims 9, 33, and 51 are believed to be allowable, and allowance of the claims is respectfully requested. Furthermore, claims 10-17, and 34-41 depend upon claims 9 and 33, respectively. Accordingly, for all of the reasons given above, Applicants submit that these dependent claims are allowable and respectfully request allowance of the same.

CONCLUSION

The Examiner errs in finding that claims 1-17, 26-41, 50-51 are anticipated by *Scalzo* under 35 U.S.C. § 102(a).

Withdrawal of the rejections and allowance of all claims is respectfully requested.

Respectfully submitted, and
S-signed pursuant to 37 CFR 1.4,

/Gero G. McClellan, 44,227/

Gero G. McClellan
Registration No. 44,227
Patterson & Sheridan, L.L.P.
3040 Post Oak Blvd., Suite 1500
Houston, TX 77056
Telephone: (713) 623-4844
Facsimile: (713) 623-4846
Attorney for Appellants